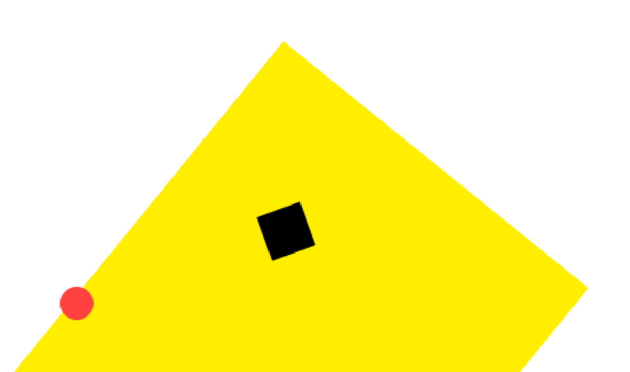
Утечки памяти в SSR

причины, поиск, устранение

Вова Захаров





Вова Захаров

- ведущий фронтенд-разработчик
- ex-Зарплата.ру
- больше 8 лет опыта в IT
- люблю плавающие баги, беседы о техдолге и шутки про ненастоящих программистов



Зарплата.ру

- федеральный джоб-борд
- около 3 млн MAU
- изоморфный код
- ReactJS, Redux, TS, nodejs, webpack
- Express на SSR
- зоопарк микросервисов на бэкенде

зарплата.ру
ТЕБЯ УСТРОИТ

Предпосылки

Как мы столкнулись с проблемой

1

Большая и долгая задача, связанная с крупной интеграцией со сторонним сервисом

2

Нет возможности тестировать и публиковать функционал по частям

3

Нет нагрузочного тестирования на SSR-сервис



1.77 GiB
1.68 GiB
1.58 GiB
1.49 GiB
1.40 GiB
1.30 GiB
1.21 GiB
1.12 GiB
1.02 GiB
954 MiB

График расхода памяти внутри контейнера SSR




График расхода памяти внутри контейнера SSR



Проблема

Хьюстон, приём!

1

SSR работает, но рестарт 2-3 раза в сутки

2

99% загрузка процессора

3

Превышена квота оперативной памяти

4

Трекер ошибок умер под внутренним DDoS



Типичный сценарий

– двигаем пиксели





Типичный сценарий

- двигаем пиксели
- пишем код для браузера





Типичный сценарий

- двигаем пиксели
- пишем код для браузера
- индивидуальная среда и ресурсы





Типичный сценарий

- двигаем пиксели
- пишем код для браузера
- индивидуальная среда и ресурсы
- малое количество действий пользователя в рамках одной сессии





Типичный сценарий

- двигаем пиксели
- пишем код для браузера
- индивидуальная среда и ресурсы
- малое количество действий пользователя в рамках одной сессии
- утечки никак себя не проявляют





Типичный сценарий

- двигаем пиксели
- пишем код для браузера
- индивидуальная среда и ресурсы
- малое количество действий пользователя в рамках одной сессии
- утечки никак себя не проявляют
- редкие случаи можно списать





Почему утечки — это проблема

на клиенте

1 видимая потеря фреймрейта и зависание UI



Почему утечки — это проблема

на клиенте

1. видимая потеря фреймрейта и зависание UI

2. повышенное потребление энергии — быстрее садится батарея



Почему утечки — это проблема

на клиенте

1 видимая потеря фреймрейта и зависание UI

2 повышенное потребление энергии — быстрее садится батарея

3 повышенная нагрузка на процессор — кто-то майнит крипту



Почему утечки — это проблема

на клиенте

1. видимая потеря фреймрейта и зависание UI
2. повышенное потребление энергии — быстрее садится батарея
3. повышенная нагрузка на процессор — кто-то майнит крипту
4. блокирование потока и падение вкладки/браузера



Почему это огромная проблема

на сервере

1 из-за высокой нагрузки утечки растут лавинообразно



Почему это огромная проблема

на сервере

1 из-за высокой нагрузки утечки растут лавинообразно





Почему это огромная проблема

на сервере

1. из-за высокой нагрузки утечки растут лавинообразно
2. тратят общие ресурсы сервера
3. проблемы одновременно начинаются у всех пользователей



Почему это огромная проблема

на сервере

1 из-за высокой нагрузки утечки растут лавинообразно

2 тратят общие ресурсы сервера

3 проблемы одновременно начинаются у всех пользователей

4 оказывают сайд-эффекты на сторонние сервисы



Как это работает



Устройство памяти V8

- объекты
- динамические данные

Heap memory

- примитивы
- ссылки
- фреймы функций
- глобальный фрейм

Stack



Работа GC в V8

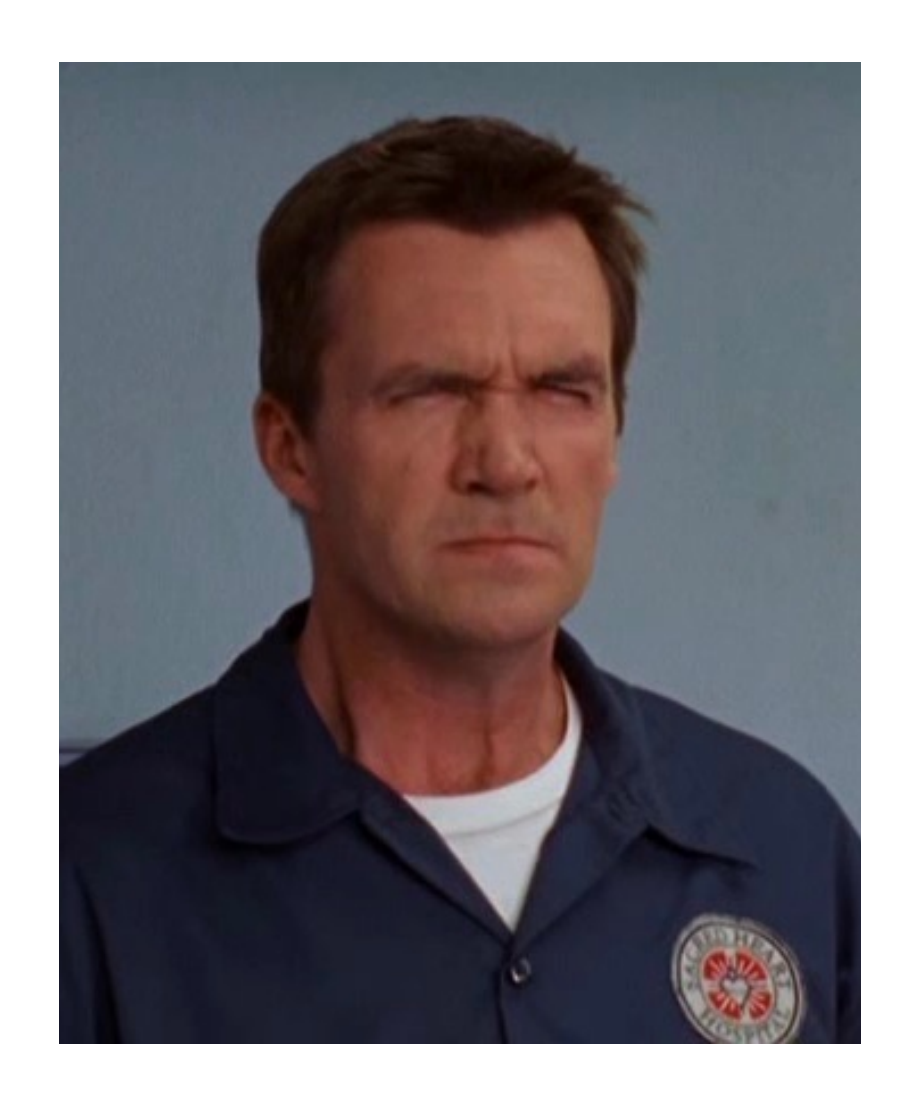
– автоматический запуск





Работа GC в V8

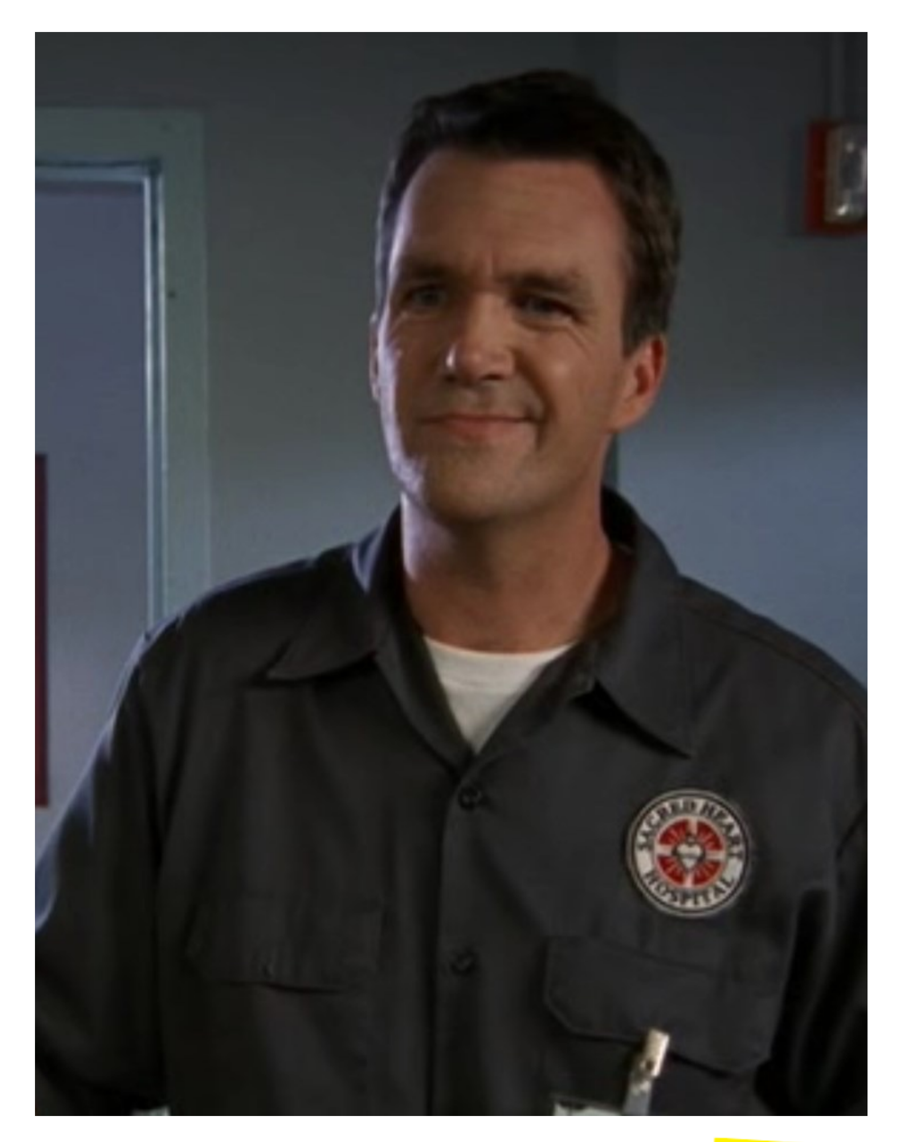
- автоматический запуск
- асинхронный и непредсказуемый





Работа GC в V8

- автоматический запуск
- асинхронный и непредсказуемый
- «необходимость» == «достижимость»





Работа GC в V8

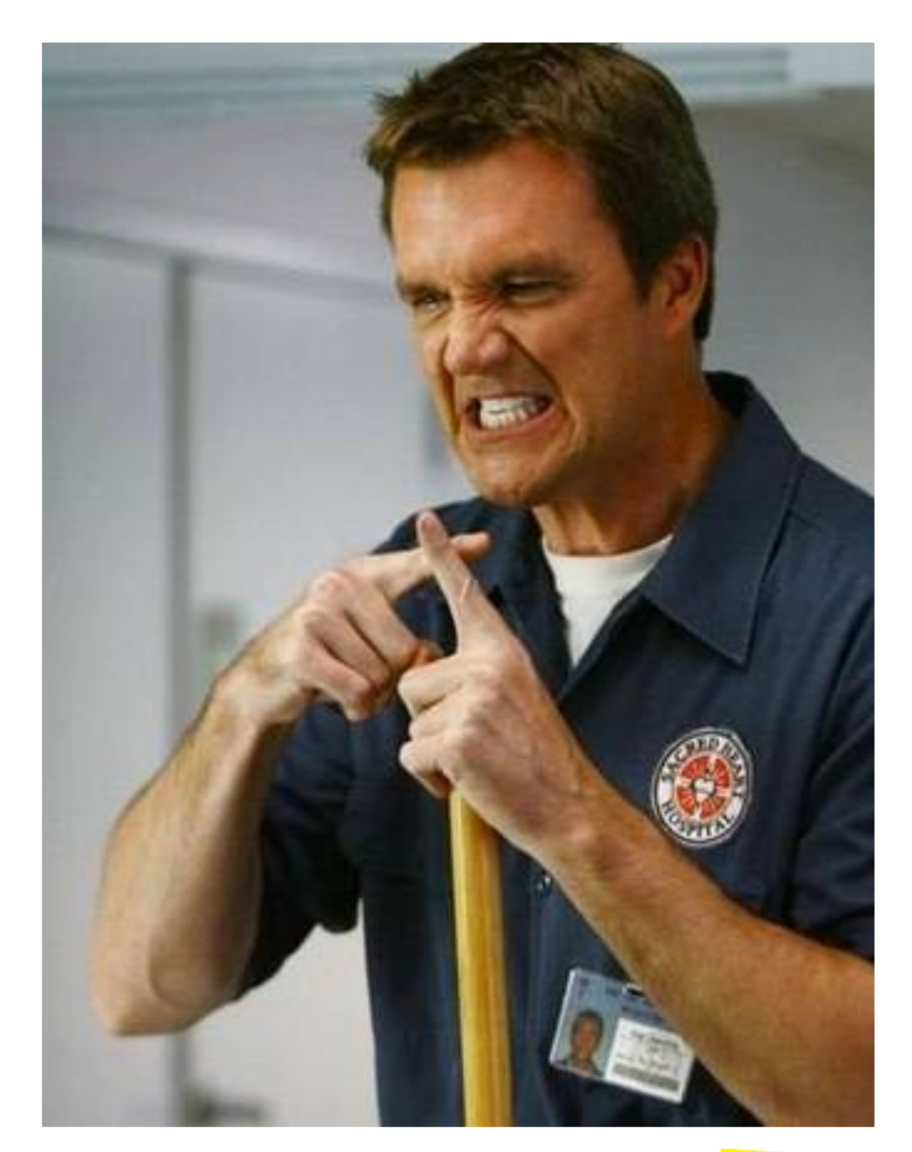
- автоматический запуск
- асинхронный и непредсказуемый
- «необходимость» == «достижимость»
- существуют слабые и сильные ссылки





Работа GC в V8

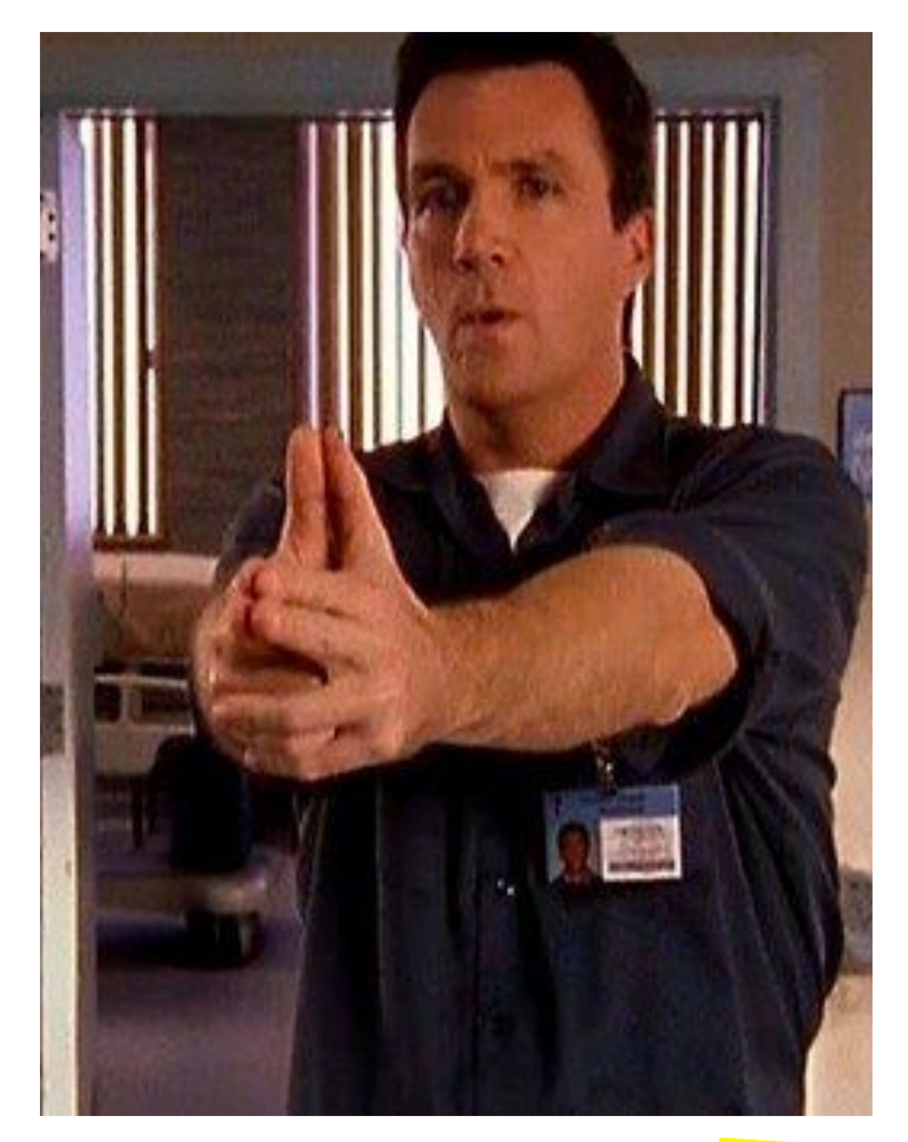
- автоматический запуск
- асинхронный и непредсказуемый
- «необходимость» == «достижимость»
- существуют слабые и сильные ссылки
- слабые ссылки создаются структурами `WeakMap`, `WeakSet`, `WeakRef`





Работа GC в V8

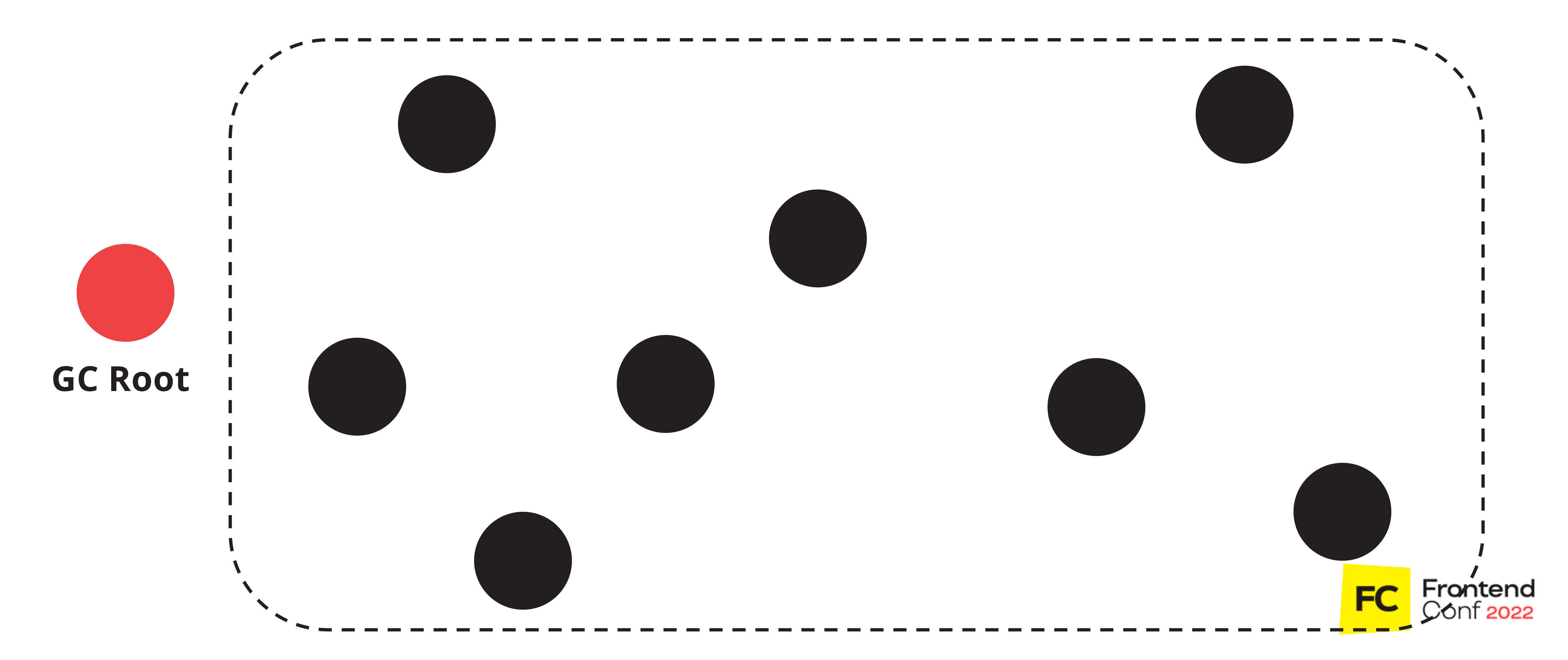
- автоматический запуск
- асинхронный и непредсказуемый
- «необходимость» == «достижимость»
- существуют слабые и сильные ссылки
- слабые ссылки создаются структурами `WeakMap`, `WeakSet`, `WeakRef`
- слабые ссылки не учитываются GC





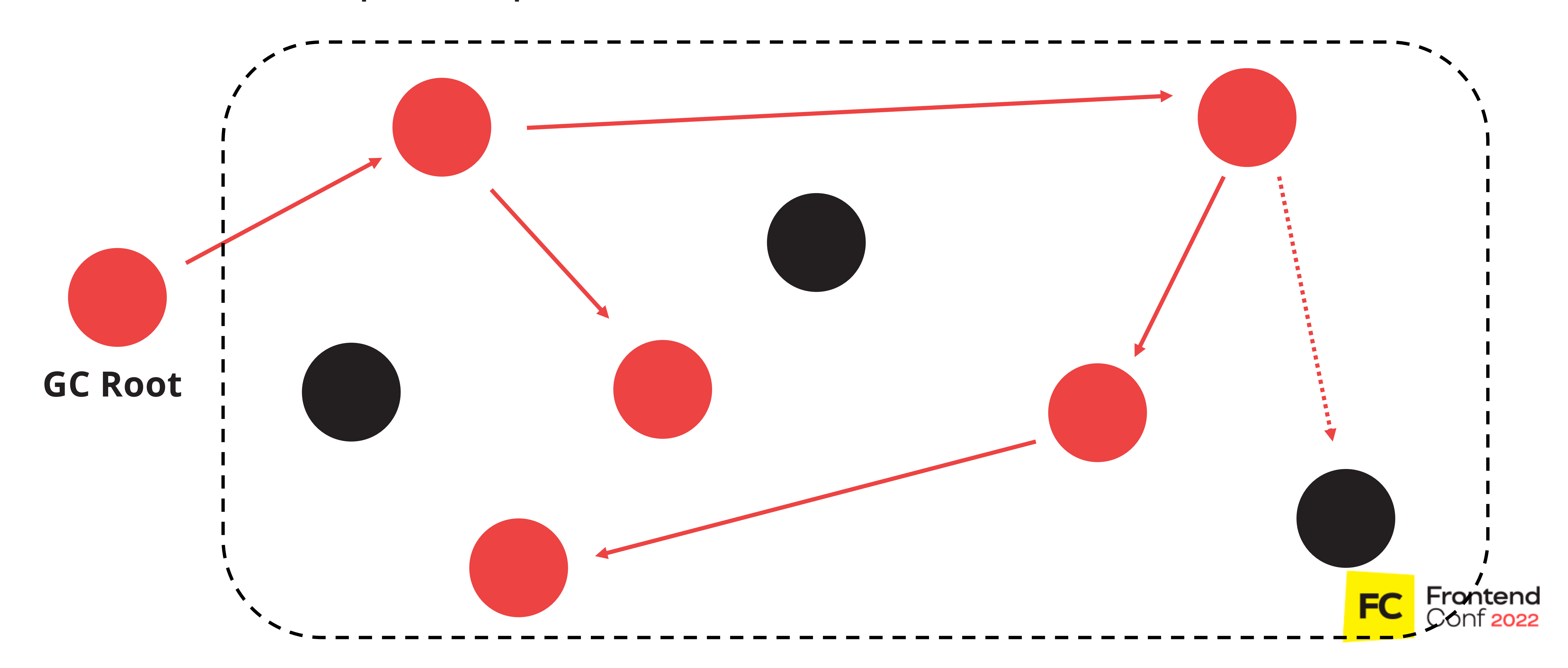
Работа GC в V8

Mark Sweep Compact



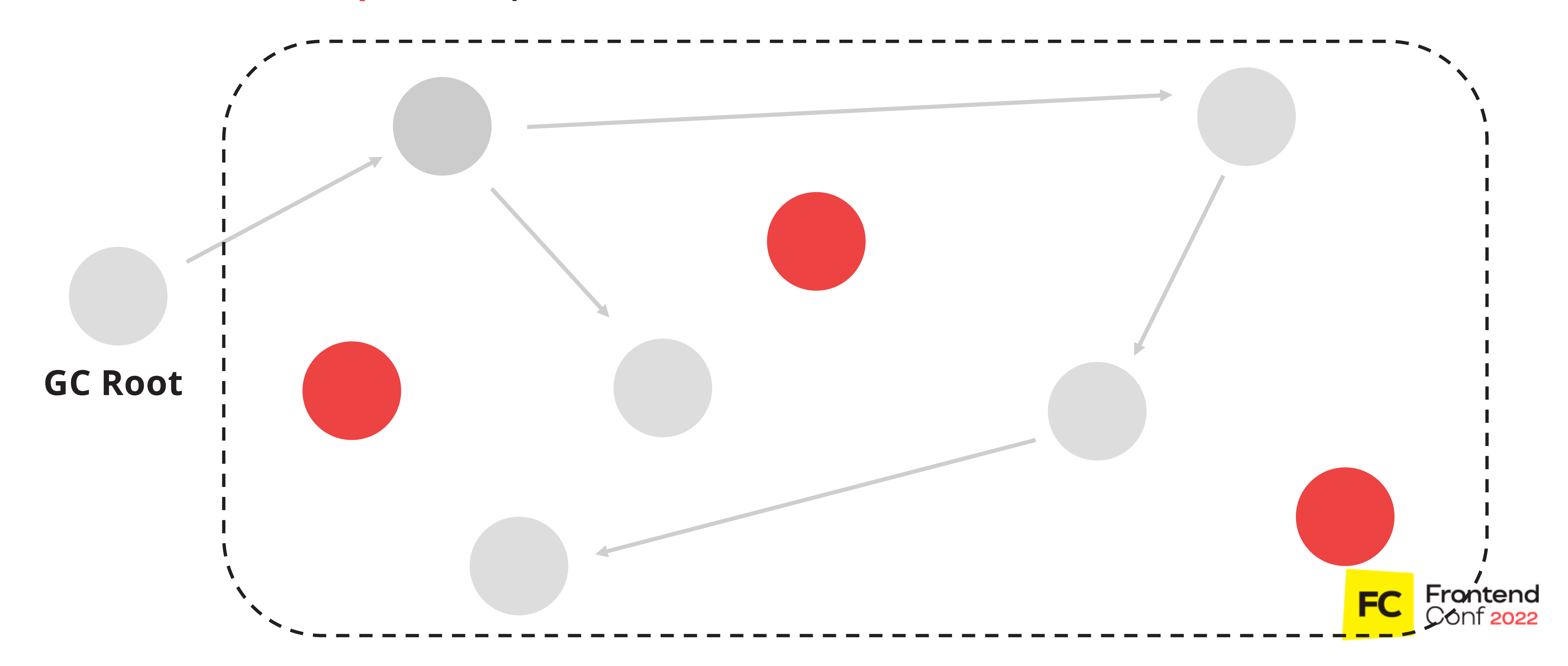
Работа GC в V8

Mark Sweep Compact



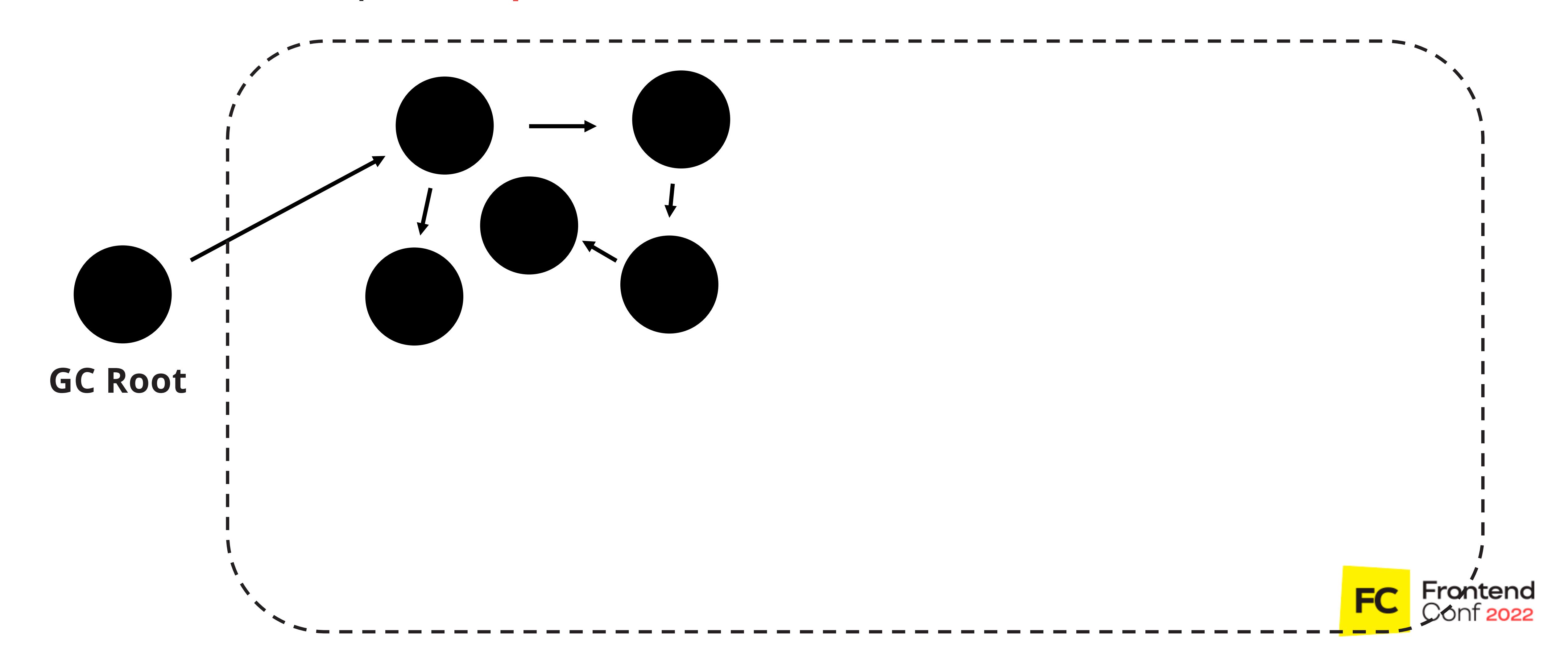
Работа GC в V8

Mark Sweep Compact



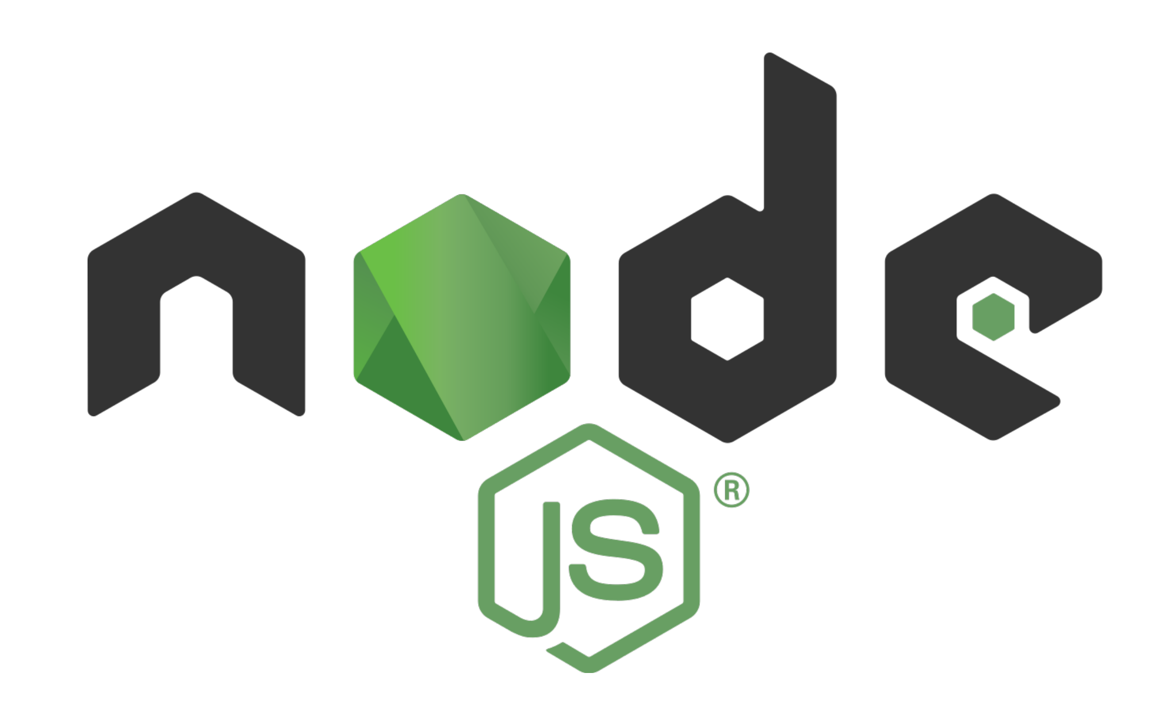
Работа GC в V8

Mark Sweep Compact



Ещё раз
о серверной среде

– один общий хип
– один общий стек
– кратный рост количества исполнений
– синглтоны, кэши, подписки, таймауты, промисы, воркеры – **всё теперь общее**





Утечка памяти

неконтролируемое уменьшение объёма свободной оперативной или виртуальной памяти компьютера, связанное с невозможностью вовремя освободить память от ненужных данных.

Утечка — накопление мусора в хипе, невозможность корректной работы GC из-за «потерянной» ссылки на объект.

ALLOCATION

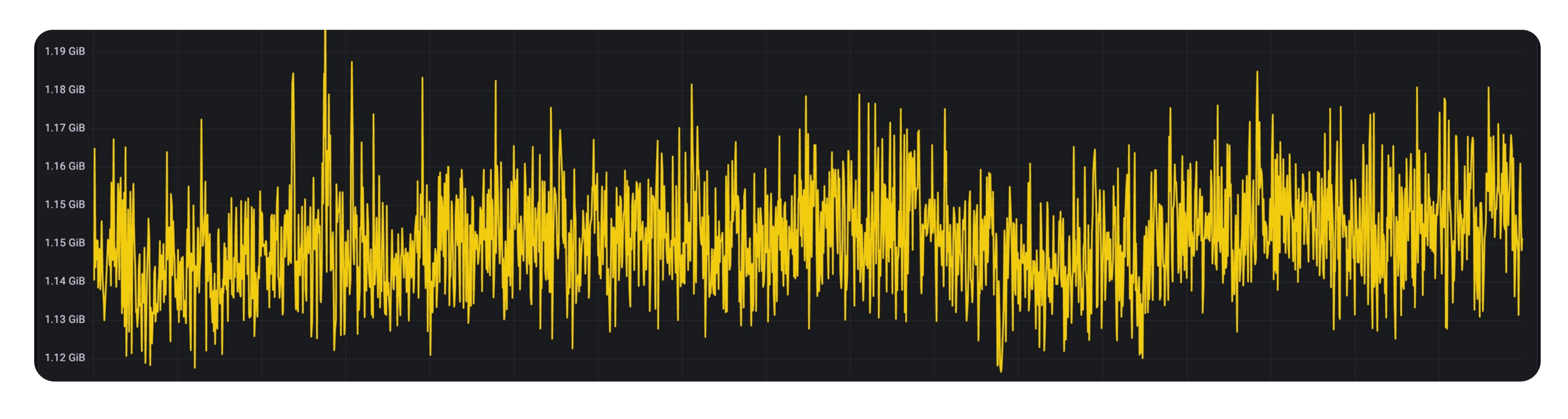


USAGE

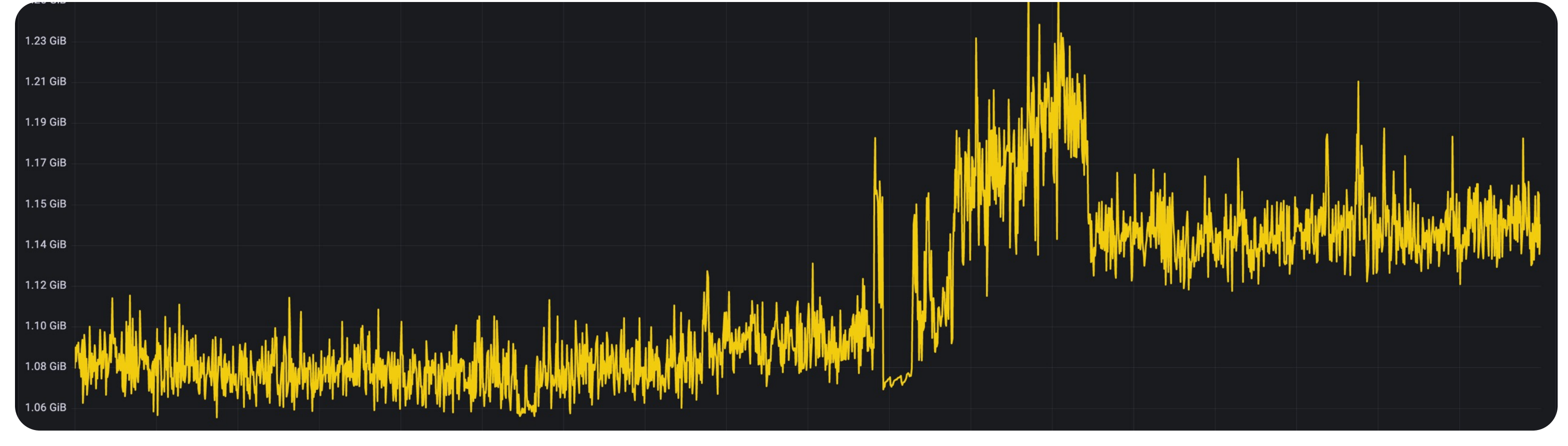




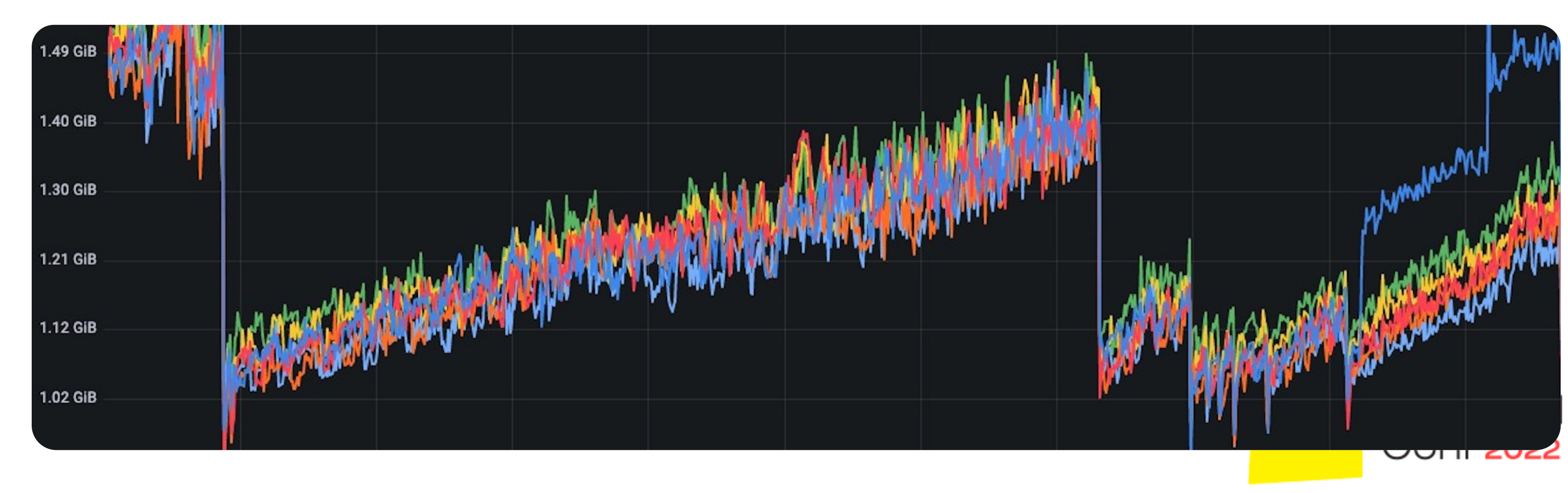
норма



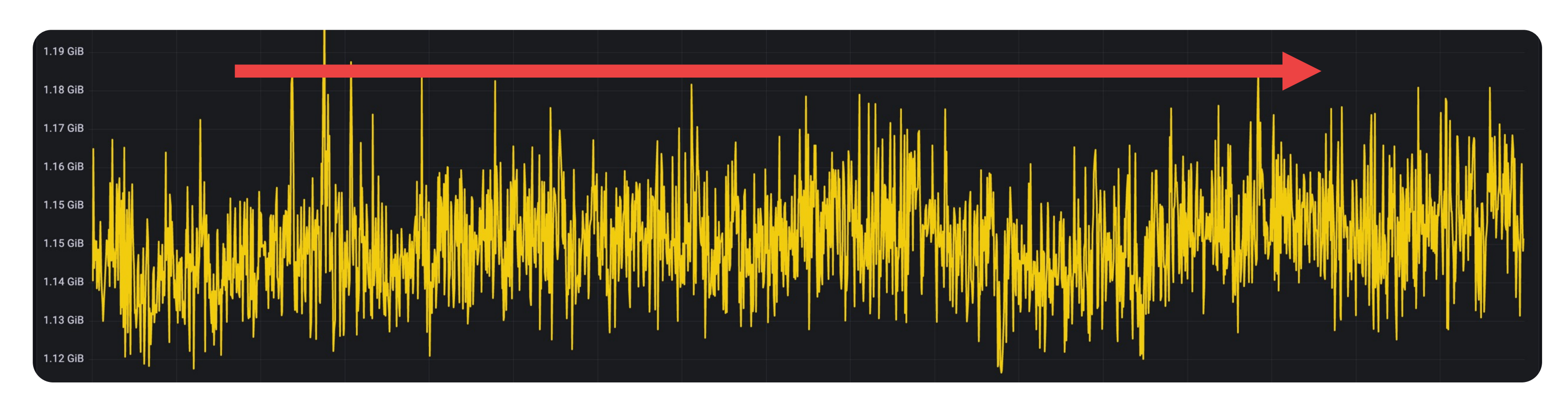
нагрузка



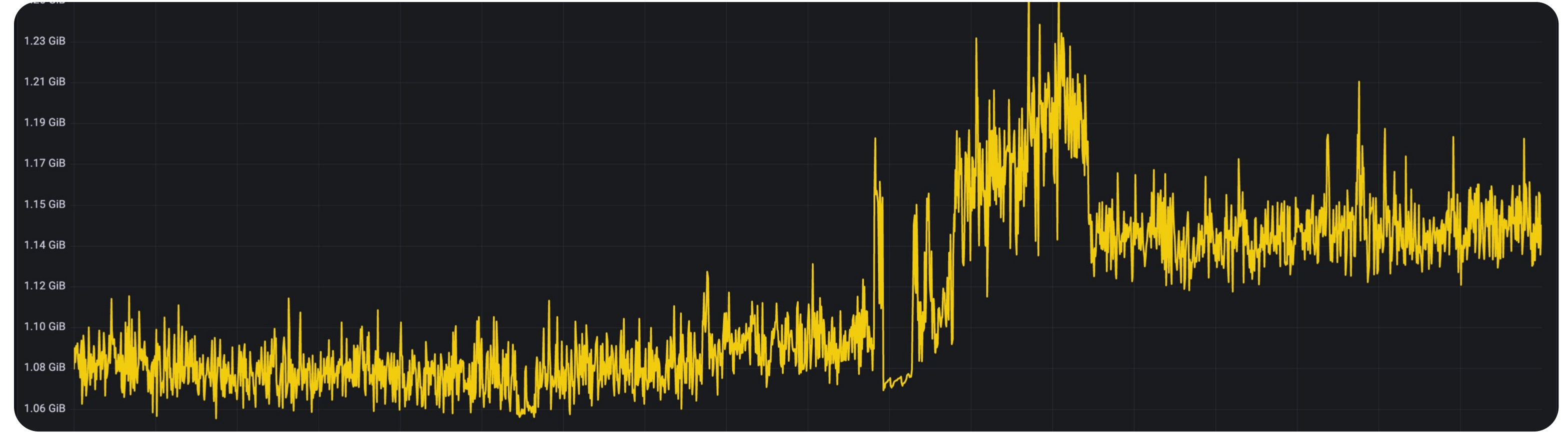
утечка



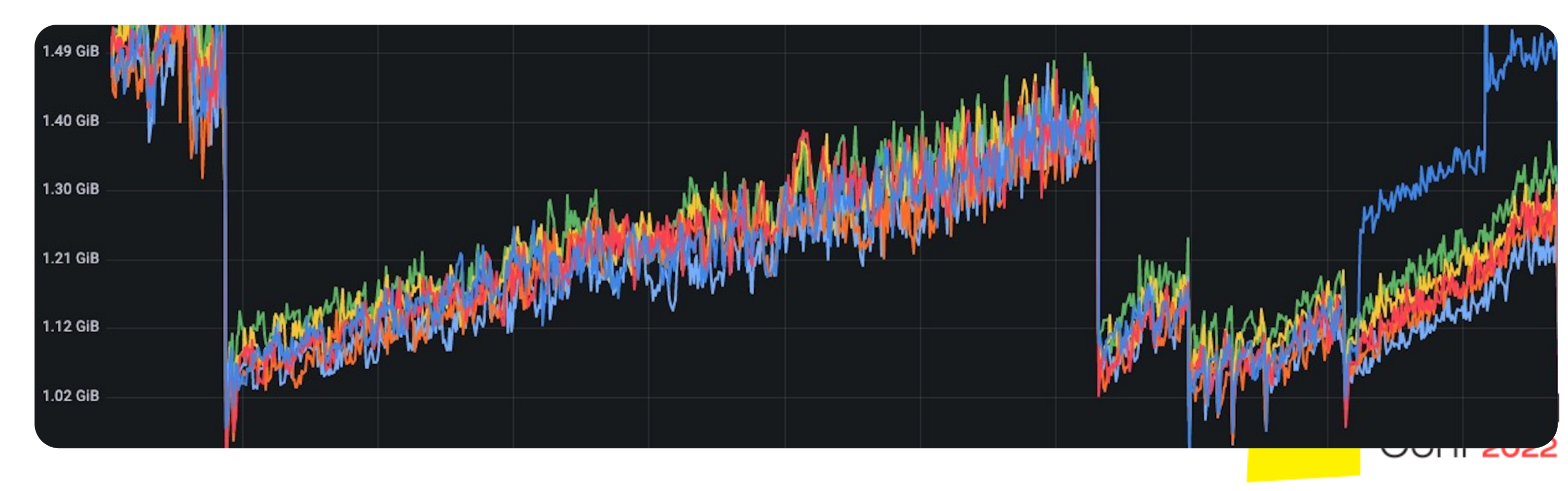
норма



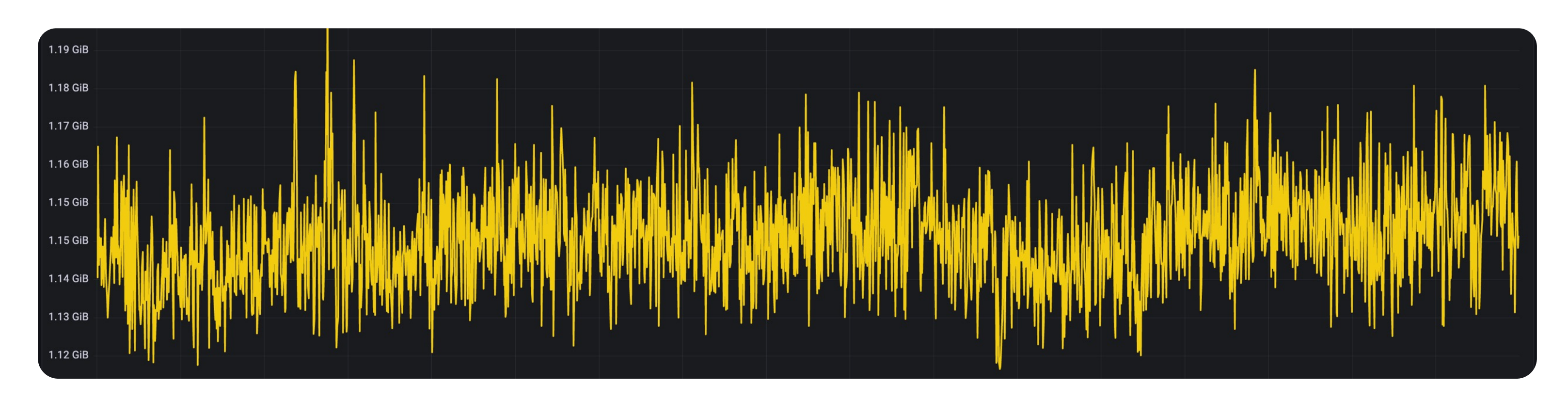
нагрузка



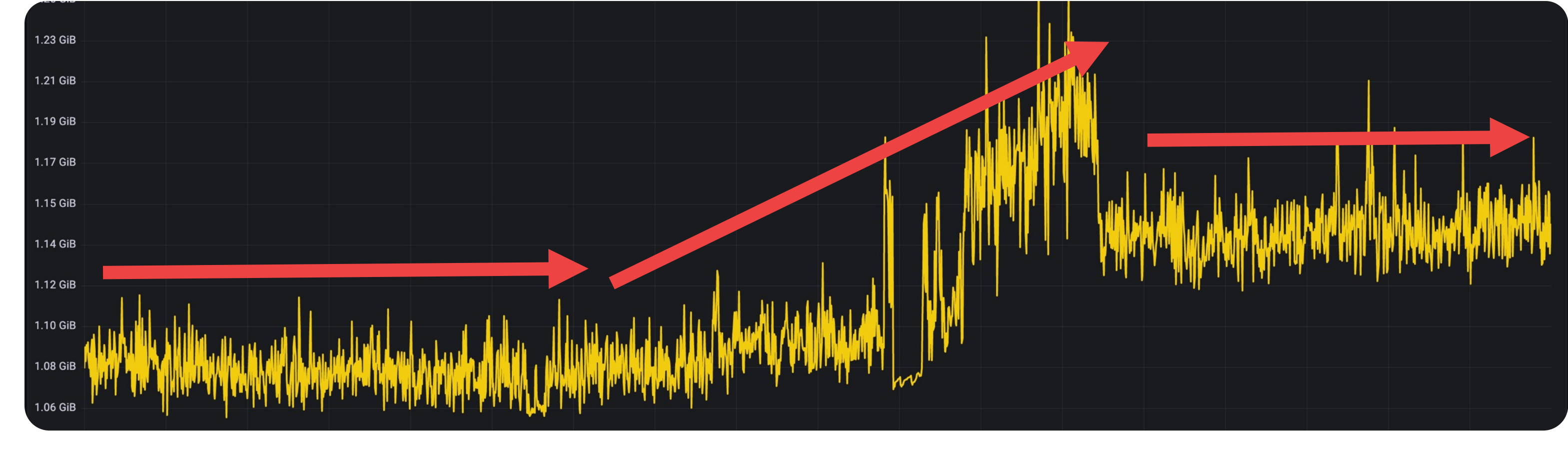
утечка



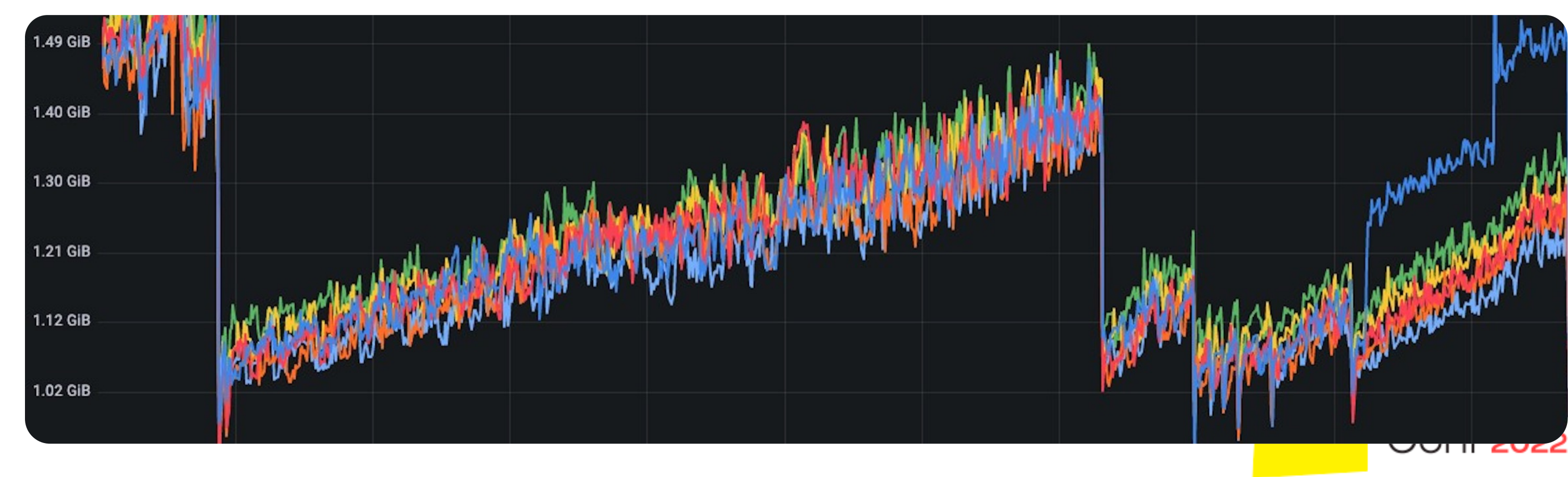
норма



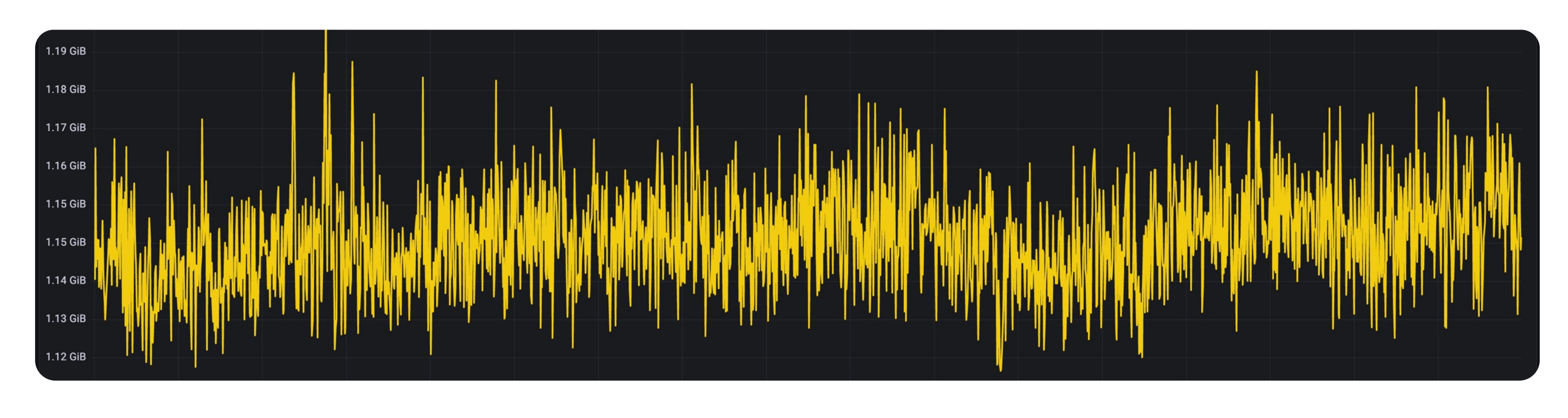
нагрузка



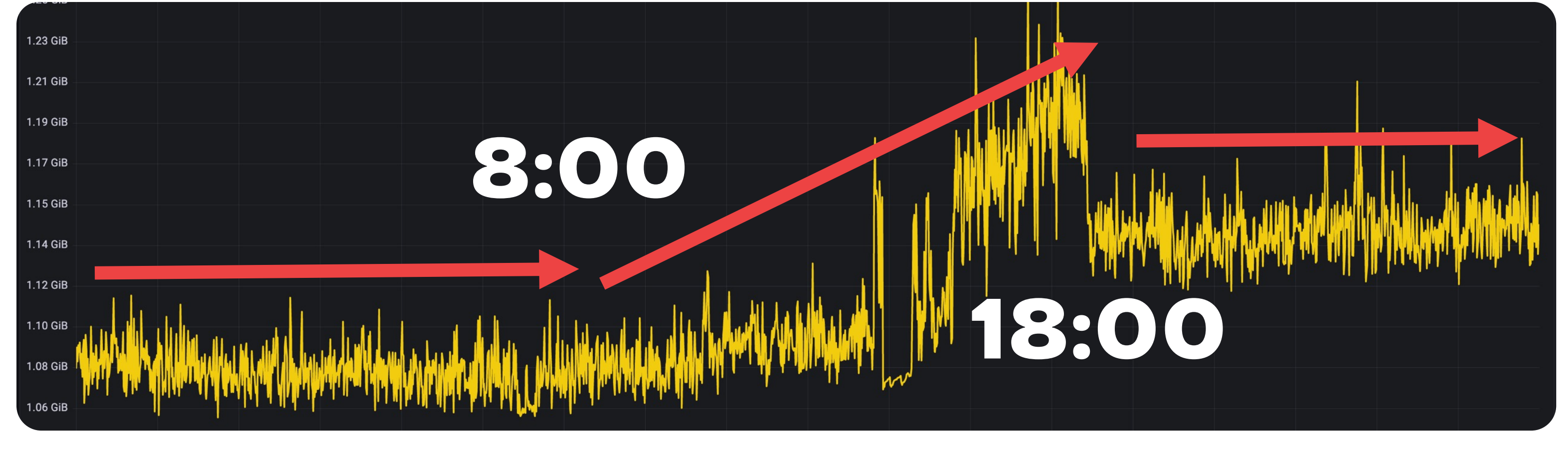
утечка



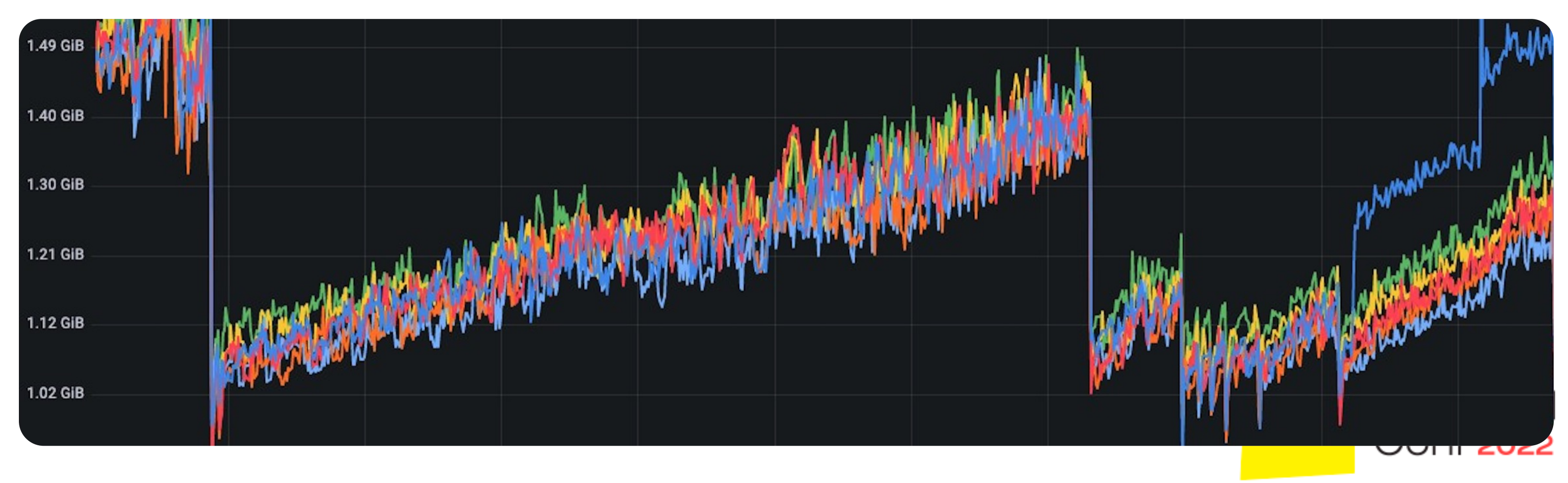
норма



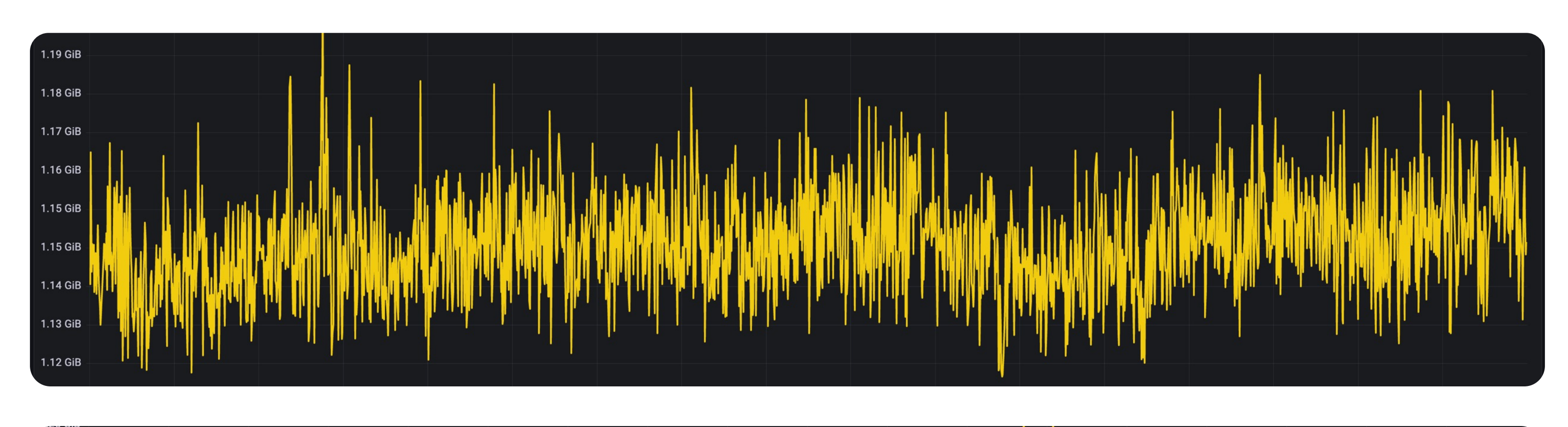
нагрузка



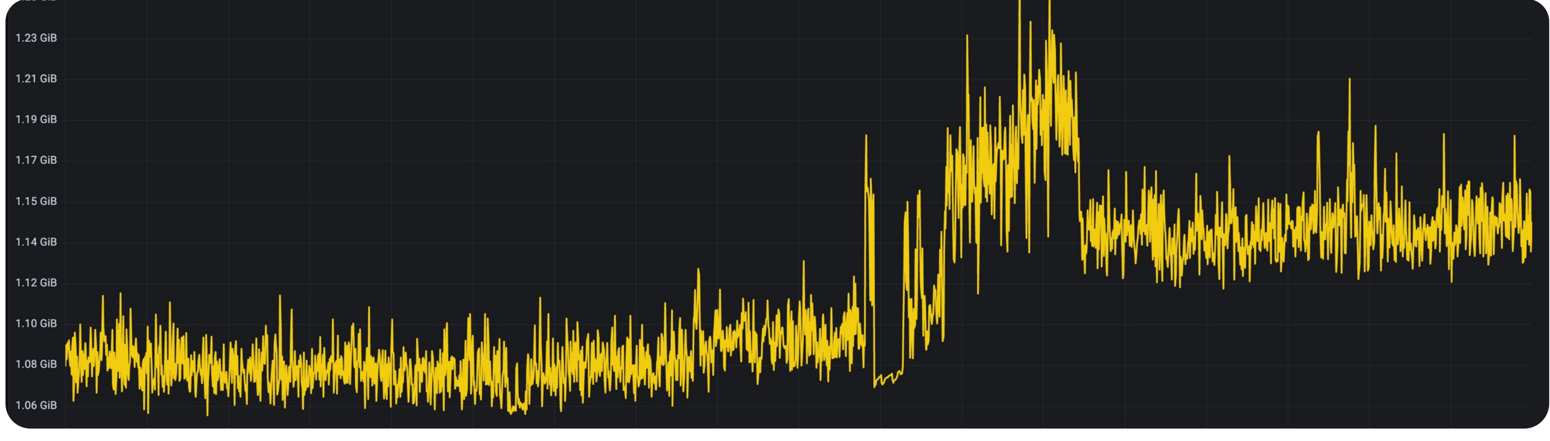
утечка



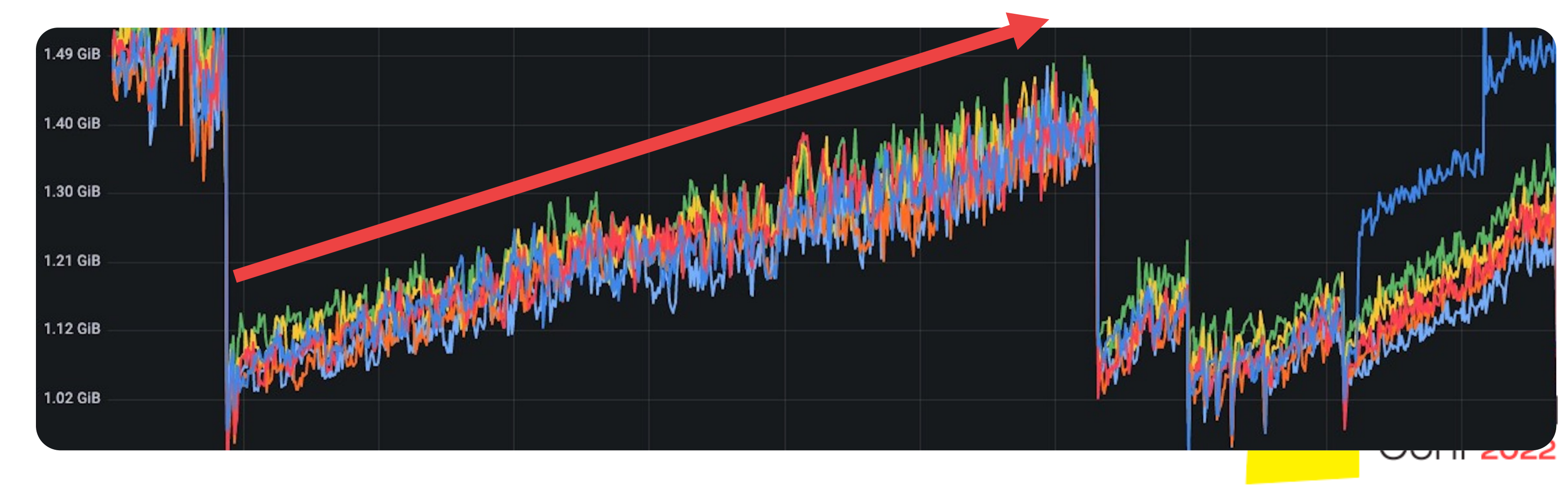
норма



нагрузка



утечка

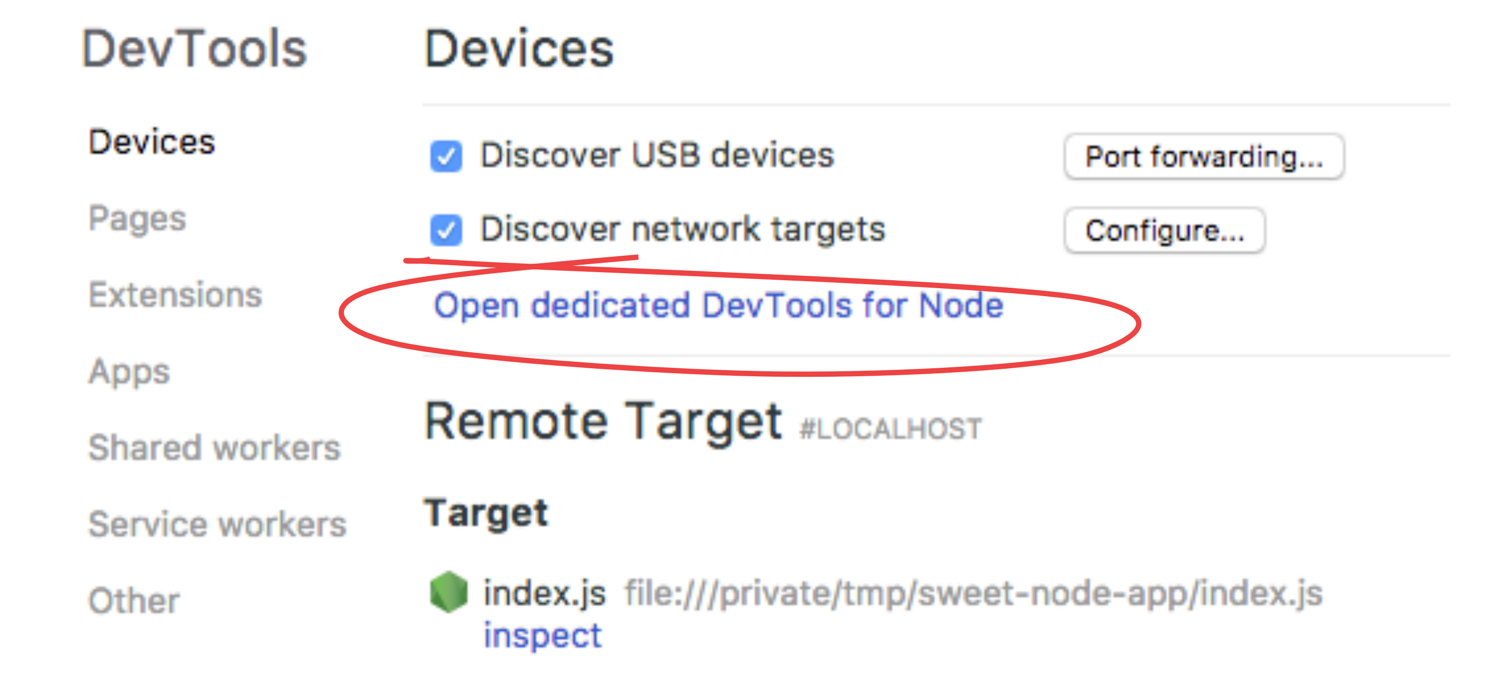


Как получить дамп с SSR?

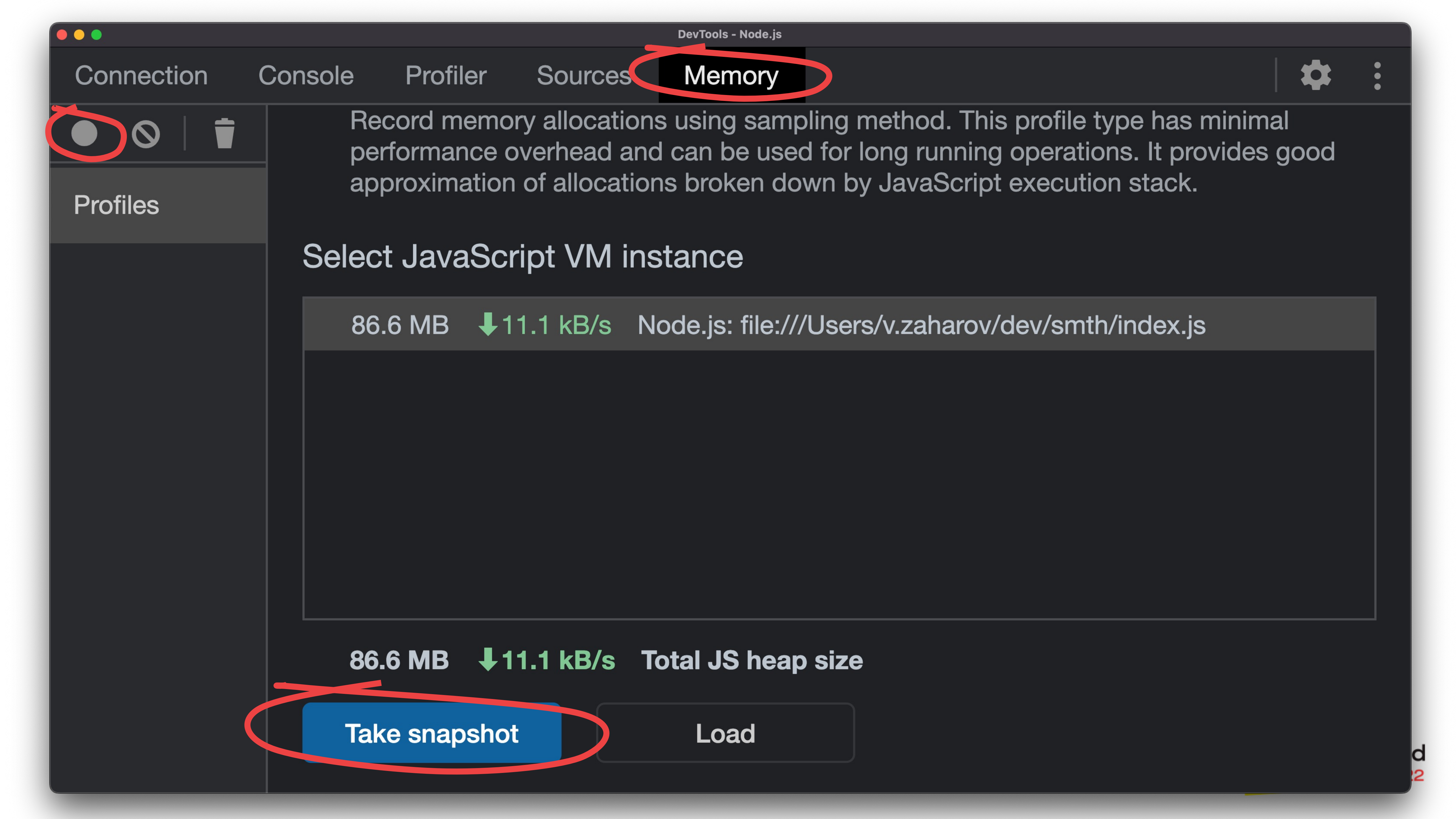


node --inspect

1. настроить port-forwarding
2. node --inspect app.js
3. chrome://inspect
4. Open dedicated DevTools for Node






DevTools - Node.js
Connection
Console
Profiler
Sources
Memory
Profiles
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.
Select JavaScript VM instance
86.6 MB
11.1 kB/s
Node.js: file:///Users/v.zaharov/dev/smth/index.js
86.6 MB
11.1 kB/s
Total JS heap size
Take snapshot
Load

Constructor	
▼ (closure) ×3896	
▶ *replaceData()* @4581	file:///Users/v.zaharov/dev/smth/index.js:43
▶ *someMethod()* @1250719	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250733	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250751	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250777	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250807	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250845	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250887	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250937	file:///Users/v.zaharov/dev/smth/index.js:52
▶ *someMethod()* @1250991	file:///Users/v.zaharov/dev/smth/index.js:52



npm heapdump

- последний релиз 3 года назад
- использует нативный nodejs—модуль
- для сборки нужен python
- из коробки не работает

```
heapdump.writeSnapshot(`./snap1.heapsnapshot`);
```



require('v8')

- для ноды версии > 11.13.0
- есть дополнительные инструменты
- работает из коробки

```
v8.writeHeapSnapshot('./snap1.heapsnapshot');
```



Как снимать снапшоты

```
setTimeout(() => {
    v8.writeHeapSnapshot(`./${Date.now()}.heapsnapshot`);
}, ONE_HOUR);
```

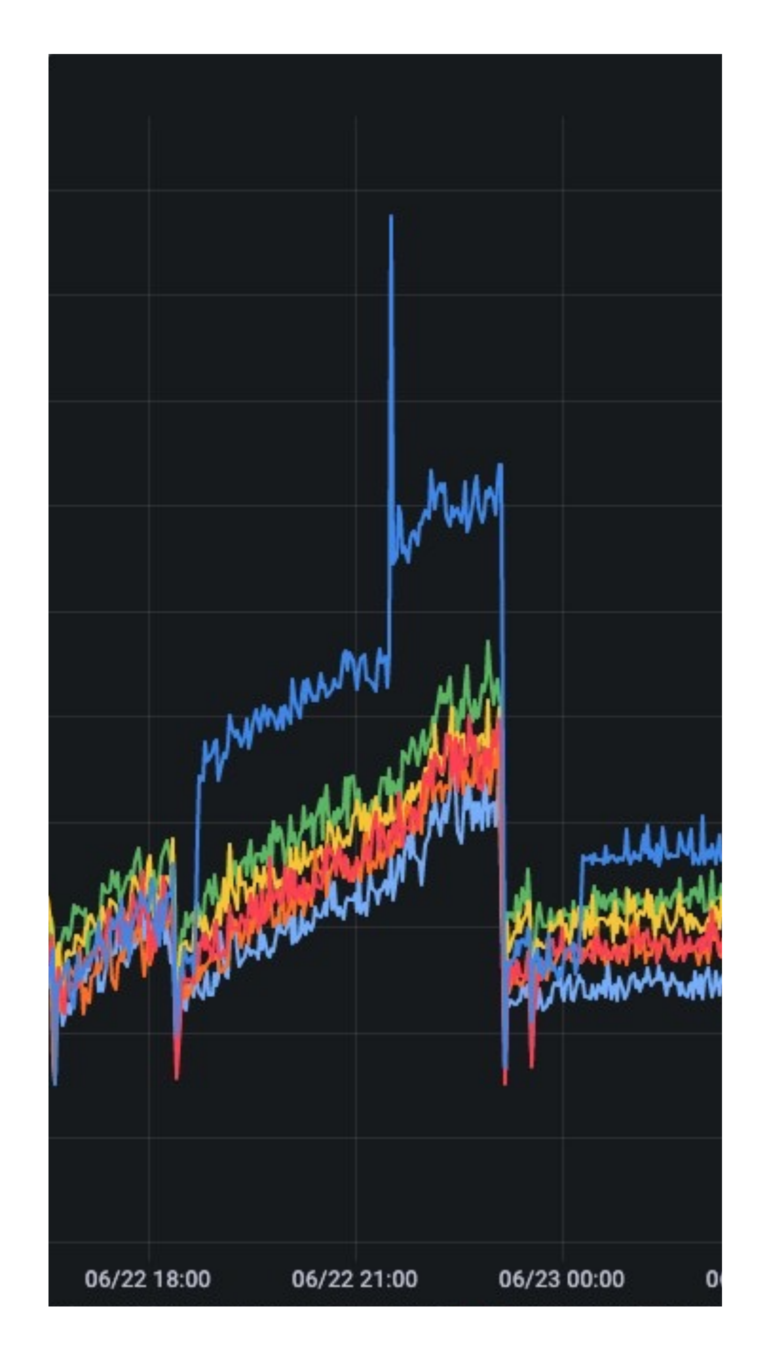
```
снять дамп вручную kill -USR2 <pid nodejs>

process.on('SIGUSR2', () => {
    v8.writeHeapSnapshot(`./${Date.now()}.heapsnapshot`);
});
```



Особенности

- объём памяти в контейнере должен быть в два раза больше, чем текущий размер хипа
- имя файла должно иметь расширение .heapsnapshot



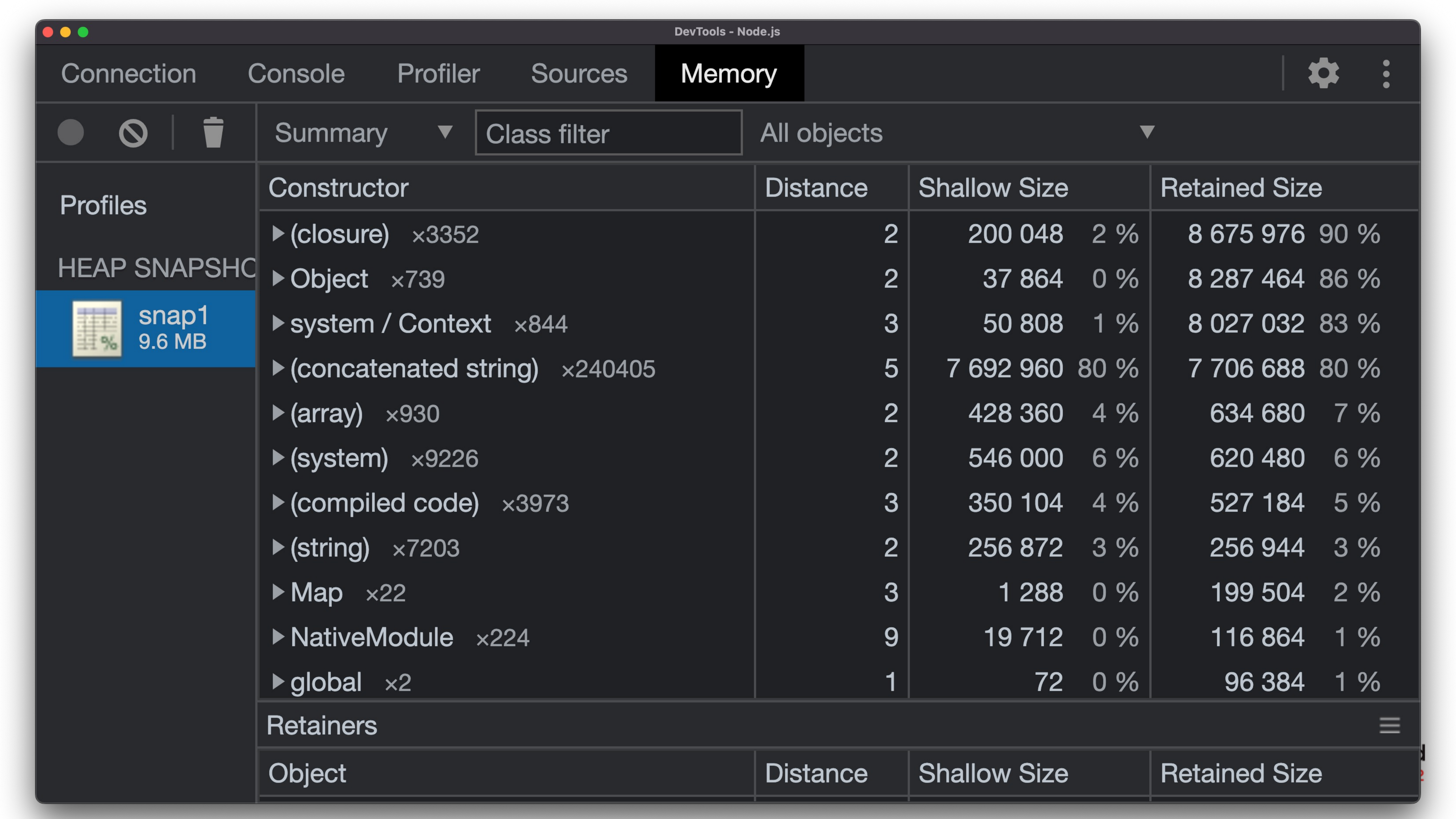


Поиск
и анализ

Chrome Dev Tools







DevTools - Node.js
Connection
Console
Profiler
Sources
Memory
Summary
Class filter
All objects
Profiles
HEAP SNAPSHO
snap1
9.6 MB
Constructor
Distance
Shallow Size
Retained Size
(closure) ×3352 2 200 048 2 % 8 675 976 90 %
Object ×739 2 37 864 0 % 8 287 464 86 %
system / Context ×844 3 50 808 1 % 8 027 032 83 %
(concatenated string) ×240405 5 7 692 960 80 % 7 706 688 80 %
(array) ×930 2 428 360 4 % 634 680 7 %
(system) ×9226 2 546 000 6 % 620 480 6 %
(compiled code) ×3973 3 350 104 4 % 527 184 5 %
(string) ×7203 2 256 872 3 % 256 944 3 %
Map ×22 3 1 288 0 % 199 504 2 %
NativeModule ×224 9 19 712 0 % 116 864 1 %
global ×2 1 72 0 % 96 384 1 %
Retainers
Object
Distance
Shallow Size
Retained Size

Profiles

HEAP SNAPSHOTS



snap1

9.6 MB



Constructor	Distance	Shallow Size		Retained Size	
▶ (closure) ×3352	2	200 048	2 %	8 675 976	90 %
▶ Object ×739	2	37 864	0 %	8 287 464	86 %
▶ system / Context ×844	3	50 808	1 %	8 027 032	83 %
▶ (concatenated string) ×24…	5	7 692 960	80 %	7 706 688	80 %
▶ (array) ×930	2	428 360	4 %	634 680	7 %
▶ (system) ×9226	2	546 000	6 %	620 480	6 %
▶ (compiled code) ×3973	3	350 104	4 %	527 184	5 %
▶ (string) ×7203	2	256 872	3 %	256 944	3 %
▶ Map ×22	3	1 288	0 %	199 504	2 %
▶ NativeModule ×224	9	19 712	0 %	116 864	1 %
▶ global ×2	1	72	0 %	96 384	1 %



Constructor	Distance	Shallow Size		Retained Size	
(closure) ×3352	2	200 048	2 %	8 675 976	90 %
Object ×739	2	37 864	0 %	8 287 464	86 %
system / Context ×844	3	50 808	1 %	8 027 032	83 %
(concatenated string) ×24…	5	7 692 960	80 %	7 706 688	80 %
(array) ×930	2	428 360	4 %	634 680	7 %
(system) ×9226	2	546 000	6 %	620 480	6 %
(compiled code) ×3973	3	350 104	4 %	527 184	5 %
(string) ×7203	2	256 872	3 %	256 944	3 %
Map ×22	3	1 288	0 %	199 504	2 %
NativeModule ×224	9	19 712	0 %	116 864	1 %
global ×2	1	72	0 %	96 384	1 %



Constructor	Distance	Shallow Size		Retained Size	
(closure) ×3352	2	200 048	2 %	8 675 976	90 %
Object ×739	2	37 864	0 %	8 287 464	86 %
system / Context ×844	3	50 808	1 %	8 027 032	83 %
(concatenated string) ×24…	5	7 692 960	80 %	7 706 688	80 %
(array) ×930	2	428 360	4 %	634 680	7 %
(system) ×9226	2	546 000	6 %	620 480	6 %
(compiled code) ×3973	3	350 104	4 %	527 184	5 %
(string) ×7203	2	256 872	3 %	256 944	3 %
Map ×22	3	1 288	0 %	199 504	2 %
NativeModule ×224	9	19 712	0 %	116 864	1 %
global ×2	1	72	0 %	96 384	1 %

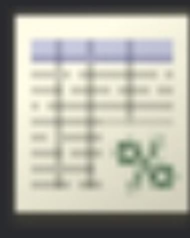


Constructor	Distance	Shallow Size		Retained Size	
(closure) ×3352	2	200 048	2 %	8 675 976	90 %
Object ×739	2	37 864	0 %	8 287 464	86 %
system / Context ×844	3	50 808	1 %	8 027 032	83 %
(concatenated string) ×24…	5	7 692 960	80 %	7 706 688	80 %
(array) ×930	2	428 360	4 %	634 680	7 %
(system) ×9226	2	546 000	6 %	620 480	6 %
(compiled code) ×3973	3	350 104	4 %	527 184	5 %
(string) ×7203	2	256 872	3 %	256 944	3 %
Map ×22	3	1 288	0 %	199 504	2 %
NativeModule ×224	9	19 712	0 %	116 864	1 %
global ×2	1	72	0 %	96 384	1 %



Profiles

HEAP SNAPSHOTS



snap1
9.6 MB



snap2
15.3 MB



snap3
24.1 MB



Constructor	Distance	Shallow Size		Retained Size	
▶ (closure) ×3943	2	237 456	1 %	23 134 176	96 %
▶ Object ×1191	2	56 128	0 %	22 714 640	94 %
▶ system / Context ×1292	3	69 760	0 %	22 466 280	93 %
▶ (concatenated string) ×68…	5	22 040 768	91 %	22 069 304	91 %
▶ (system) ×9894	2	582 392	2 %	663 600	3 %
▶ (array) ×1026	2	438 304	2 %	649 120	3 %
▶ (compiled code) ×4117	3	358 456	1 %	551 384	2 %
▶ (string) ×7913	2	280 456	1 %	280 528	1 %
▶ Map ×22	3	1 288	0 %	205 160	1 %
▶ NativeModule ×224	9	19 712	0 %	112 792	0 %

Summary ▼ | Class filter

- ✓ All objects
- Objects allocated before snap1
- Objects allocated between snap1 and snap2
- Objects allocated between snap2 and snap3

Profiles

HEAP SNAPSHOTS

- snap1 9.6 MB
- snap2 15.3 MB
- snap3 24.1 MB

Constructor	Dista				
(closure) ×3943					
Object ×1191	2	56 128	0 %	22 714 640	94 %
system / Context ×1292	3	69 760	0 %	22 466 280	93 %
(concatenated string) ×688774	5	22 040 768	91 %	22 069 304	91 %
(system) ×9894	2	582 392	2 %	663 600	3 %
(array) ×1026	2	438 304	2 %	649 120	3 %
(compiled code) ×4117	3	358 456	1 %	551 384	2 %

- All objects
- Objects allocated before Snapshot 1
- ✓ Objects allocated between Snapshot 1 and Snapshot 2
- Objects allocated between Snapshot 2 and Snapshot 3



Summary ▼ | Class filter | Objects allocated between snap1 and snap2 ▼

Constructor	Distance	Shallow Size		Retained Size ▼	
▶ Object ×184	6	7 592	0 %	13 261 848	55 %
▶ (closure) ×319	4	20 000	0 %	13 250 464	55 %
▶ system / Context ×175	7	8 032	0 %	13 230 928	55 %
▶ (concatenated string) ×172093	6	5 506 976	23 %	5 513 048	23 %
▶ (system) ×818	5	47 120	0 %	53 888	0 %
▶ (compiled code) ×146	5	8 464	0 %	19 240	0 %
▶ (array) ×120	5	16 384	0 %	17 832	0 %
▶ (string) ×437	5	14 848	0 %	14 848	0 %



Constructor	Distance	Shallow Size		Retained Size ▼	
▶ Object ×184	6	7 592	0 %	13 261 848	55 %
▼ (closure) ×319	4	20 000	0 %	13 250 464	55 %
▶ *someMethod() @871825*	829	64	0 %	13 219 464	55 %
▶ *someMethod() @839899*	832	64	0 %	13 186 904	55 %
▶ *someMethod() @833215*	835	64	0 %	13 154 344	55 %
▶ *someMethod() @831183*	838	64	0 %	13 121 784	54 %
▶ *someMethod() @829151*	841	64	0 %	13 089 224	54 %
▶ *someMethod() @829139*	844	64	0 %	13 056 664	54 %
▶ *someMethod() @879969*	847	64	0 %	13 024 104	54 %
▶ *someMethod() @877937*	850	64	0 %	12 991 544	54 %
▶ *someMethod() @875905*	853	64	0 %	12 958 984	54 %

Паттерны утечек



Глобальные переменные

```
function foo() {
    bar = { val: "hello world" };
}
```

```
function foo() {
    this.bar = { val: "hello world" };
}

foo();
```

```
<script>
    var bar = { val: "hello world" };
</script>
```

```
global.bar.val === "hello world"
```



Интервалы, таймеры, подписки

```
function handleInterval() {
    const hugeArray = [ ... ];
    const newArray = hugeArray.map(() => { ... });
}

setInterval(handleInterval, 1000);
```



Интервалы, таймеры, подписки

критерий очистки таймера / интервала

```
function handleInterval() {
    const hugeArray = [ ... ];
    const newArray = hugeArray.map(() ⇒ { ... });
}

const intervalID = setInterval(handleInterval, 1000);

// ...

if ( ... ) {
    clearInterval(intervalID);
}
```



Кэш и мемоизация

```
import memoize from 'lodash/memoize';

export const generateCompanyUrl = memoize(
    (company: object) => {
        // ...
        return companyUrl;
    }
);
```



Кэш и мемоизация

```
import memoize from 'lodash/memoize';

export const generateCompanyUrl = memoize(
    (company: object) => {
        // ...
        return companyUrl;
    }
);
```



Кэш и мемоизация

```
import memoize from 'lodash/memoize';

export const generateCompanyUrl = memoize(
    (company: object) ⇒ {
        // ...
        return companyUrl;
    }
);
```

– не кэшируйте всё подряд



Кэш и мемоизация

```
import memoize from 'lodash/memoize';

export const generateCompanyUrl = memoize(
    (company: object) => {
        // ...
        return companyUrl;
    }
);
```

- не кэшируйте всё подряд
- не используйте объекты как ключи



Кэш и мемоизация

```
import memoize from 'lodash/memoize';

export const generateCompanyUrl = memoize(
    (company: object) => {
        // ...
        return companyUrl;
    }
);
```

- не кэшируйте всё подряд
- не используйте объекты как ключи
- **lodash memoize** хранит ссылки вечно



Кэш и мемоизация

```
import memoize from 'lodash/memoize';

export const generateCompanyUrl = memoize(
    (company: object) => {
        // ...
        return companyUrl;
    }
);
```

- не кэшируйте всё подряд
- не используйте объекты как ключи
- **lodash memoize** хранит ссылки вечно
- используйте `WeakMap, WeakSet`



```
let data = null;

function replaceData() {
  const previosData = data;

  function unusedMethod() {
    if (previosData) {}
  };

  data = {
    id: generateId(1024),
    someMethod: function () {},
  };
};

setInterval(replaceData, 100);
```

Замыкания



```
let data = null;

function replaceData() {
  const previosData = data;

  function unusedMethod() {
    if (previosData) {}
  };

  data = {
    id: generateId(1024),
    someMethod: function () {},
  };
};

setInterval(replaceData, 100);
```

Замыкания



```
let data = null;

function replaceData() {
  const previosData = data;

  function unusedMethod() {
    if (previosData) {}
  };

  data = {
    id: generateId(1024),
    someMethod: function () {},
  };
};

setInterval(replaceData, 100);
```

Замыкания



Constructor	Distance	Shallow Size		Retained Size ▼	
▶ Object ×184	6	7 592	0 %	13 261 848	55 %
▼ (closure) ×319	4	20 000	0 %	13 250 464	55 %
▶ *someMethod()* @871825	829	64	0 %	13 219 464	55 %
▶ *someMethod()* @839899	832	64	0 %	13 186 904	55 %
▶ *someMethod()* @833215	835	64	0 %	13 154 344	55 %
▶ *someMethod()* @831183	838	64	0 %	13 121 784	54 %
▶ *someMethod()* @829151	841	64	0 %	13 089 224	54 %
▶ *someMethod()* @829139	844	64	0 %	13 056 664	54 %
▶ *someMethod()* @879969	847	64	0 %	13 024 104	54 %
▶ *someMethod()* @877937	850	64	0 %	12 991 544	54 %
▶ *someMethod()* @875905	853	64	0 %	12 958 984	54 %

Constructor	D.	Shallow…		Retained Size ▼	
▼(closure) ×3671	2	040	1 %	14 282 544	93 %
▼*someMethod() @871825*	10	64	0 %	13 219 464	86 %
▶context :: system / Context	11	40	0 %	13 219 400	86 %
▶__proto__ :: () @8851 ▯	3	56	0 %	920	0 %
▶map :: system / Map @9229	3	72	0 %	216	0 %
▶shared :: someMethod @34337	11	56	0 %	144	0 %
▶code :: (CompileLazy builti	3	128	0 %	128	0 %
▶feedback_cell :: system @34	11	24	0 %	24	0 %



```
▼context :: system / Context @871821
  ▶previous :: system / Context @4581
  ▼previosData :: Object @839895
    ▶someMethod :: someMethod() @839899
    ▶id :: "fhdQXQVMjKMIuigDCOWstfwaEY1XUQqGZOiJ
    ▶map :: system / Map @34347
    ▶__proto__ :: Object @9119 ▯
  ▶scope_info :: (function scope info)[] @34359
```



```
let data = null;

function replaceData() {
  const previosData = data;

  function unusedMethod() {
    if (previosData) {}
  };

  data = {
    id: generateId(1024),
    someMethod: function () {},
  };
};

setInterval(replaceData, 100);
```

Замыкания



```
let data = null;

function replaceData() {
  const previosData = data;

  function unusedMethod() {
    if (previosData) {}
  };

  data = {
    id: generateId(1024),
    someMethod: function () {},
  };
};

setInterval(replaceData, 100);
```

Замыкания

- лексическое окружение едино для всех создаваемых функций



Подстроки

```
function createString() {
  return "0".repeat(25 * 1024 * 1024).substring(0, 12);
}

const arr = [];

setInterval(function() {
   const str = createString();
   arr.push(str);
}, 500);
```



Summary ▼ Class filter Objects allocated between Snapshot 1 and Snapshot 2 ▼

Profiles

HEAP SNAPSHOTS

- Snapshot 1 — 1.0 MB
- Snapshot 2 — 1.0 MB
- Snapshot 3 — 1.0 MB — Save

Constructor	Distance	Shallow Size		Retained Size	
▶ (compiled code) ×5	4	192	0 %	248	0 %
▼ (string) ×7	3	168	0 %	168	0 %
▶ "000000000000" @52127	3	24	0 %	24	0 %
▶ "000000000000" @52129	3	24	0 %	24	0 %
▶ "000000000000" @52131	3	24	0 %	24	0 %
▶ "000000000000" @52133	3	24	0 %	24	0 %
▶ "000000000000" @52135	3	24	0 %	24	0 %
▶ "000000000000" @52137	3	24	0 %	24	0 %
▶ "000000000000" @52139	3	24	0 %	24	0 %
▶ (object shape) ×2	4	36	0 %	36	0 %

Retainers

Object	Distance	Shallow Size	Retained Size

Дамп памяти для подстрок длиной 12 символов, вкладка Memory в Chrome Dev Tools



Подстроки

начиная с длины в 13 символов, хранят ссылку на родительскую строку

```
function createString() {
  return "0".repeat(25 * 1024 * 1024).substring(0, 13);
}

const arr = [];

setInterval(function() {
   const str = createString();
   arr.push(str);
}, 500);
```



Elements | Console | Performance insights | Recorder | Lighthouse | Network | Redux | Sources | Performance | Memory | Application | Security

Summary ▼ | Class filter | Objects allocated between Snapshot 1 and Snapshot 2 ▼

Profiles

HEAP SNAPSHOTS

- Snapshot 1 — 132 MB
- Snapshot 2 — 289 MB
- Snapshot 3 — 447 MB — Save

Constructor	Distance	Shallow Size		Retained Size	
▼(sliced string) ×6	3	120	0 %	157 286 592	35 %
▼(sliced string) @60701	3	20	0 %	26 214 432	6 %
▶parent :: "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000	4	26 214 412	6 %	26 214 412	6 %
▶map :: system / Map (SlicedOneByteString) @239	4	40	0 %	40	0 %
▶(sliced string) @60705	3	20	0 %	26 214 432	6 %
▶(sliced string) @60709	3	20	0 %	26 214 432	6 %
▶(sliced string) @60713	3	20	0 %	26 214 432	6 %
▶(sliced string) @60717	3	20	0 %	26 214 432	6 %
▶(sliced string) @60719	3	20	0 %	26 214 432	6 %
▼(string) ×6	4	157 286 472	35 %	157 286 472	35 %
▶"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000	4	26 214 412	6 %	26 214 412	6 %
▶"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000	4	26 214 412	6 %	26 214 412	6 %
▶"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000	4	26 214 412	6 %	26 214 412	6 %
▶"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000	4	26 214 412	6 %	26 214 412	6 %

Retainers

Object	Distance	Shallow Size	Retained Size

Дамп памяти для подстрок длиной 13 символов, вкладка Memory в Chrome Dev Tools



Подстроки

начиная с длины в 13 символов, хранят ссылку на родительскую строку

```
function createString() {
  return "0".repeat(25 * 1024 * 1024).substring(0, 13);
}

const arr = [];

setInterval(function() {
   const str = createString().split('').join('');
   arr.push(str);
}, 500);
```



Как дальше жить?





Что делать?

- мониторинг потребления ресурсов



Что делать?

- мониторинг потребления ресурсов
- автоматические оповещения при превышении квоты



Что делать?

- мониторинг потребления ресурсов
- автоматические оповещения при превышении квоты
- понимание принципа общего использования ресурсов на Server side



Что делать?

- мониторинг потребления ресурсов
- автоматические оповещения при превышении квоты

- понимание принципа общего использования ресурсов на Server side
- своевременное обновление пакетов



Что делать?

- мониторинг потребления ресурсов
- автоматические оповещения при превышении квоты

- понимание принципа общего использования ресурсов на Server side
- своевременное обновление пакетов
- преждевременная оптимизация — зло



Что делать?

- мониторинг потребления ресурсов
- автоматические оповещения при превышении квоты

- понимание принципа общего использования ресурсов на Server side
- своевременное обновление пакетов
- преждевременная оптимизация — зло
- учёт возможности утечки при написании кода



Что делать?

- мониторинг потребления ресурсов
- автоматические оповещения при превышении квоты

- понимание принципа общего использования ресурсов на Server side
- своевременное обновление пакетов
- преждевременная оптимизация — зло
- учёт возможности утечки при написании кода

- понять и простить, мы же фронтендеры



Summary

- принципы работы сборщика мусора





Summary

- принципы работы сборщика мусора
- особенности серверной среды





Summary

- принципы работы сборщика мусора
- особенности серверной среды
- способы получения дампа памяти





Summary

- принципы работы сборщика мусора
- особенности серверной среды
- способы получения дампа памяти
- устройство вкладки Memory в ChDT





Summary

- принципы работы сборщика мусора
- особенности серверной среды
- способы получения дампа памяти
- устройство вкладки Memory в ChDT
- метод поиска и анализа утечек в дампе





Summary

- принципы работы сборщика мусора
- особенности серверной среды
- способы получения дампа памяти
- устройство вкладки Memory в ChDT
- метод поиска и анализа утечек в дампе
- паттерны утечек на сервере





Спасибо за внимание!

Остались вопросы?

tg: @vzkhrv



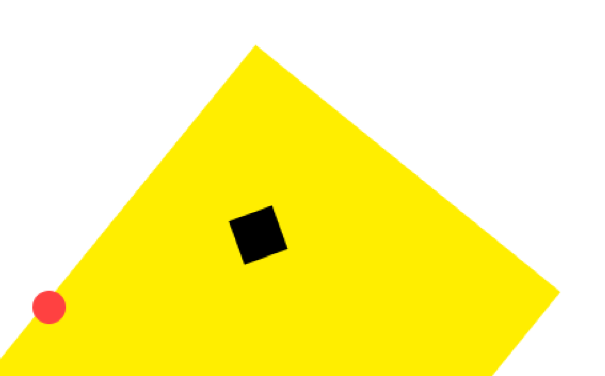
Контур



- мониторинг ресурсов
- автоматические оповещения
- принцип общей среды на Server side
- обновление пакетов
- преждевременная оптимизация — зло
- знать паттерны утечек





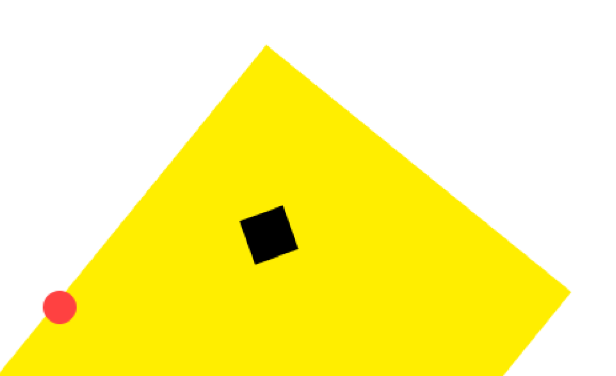


tg: @vzkhrv

- мониторинг ресурсов
- автоматические оповещения
- принцип общей среды на Server side
- обновление пакетов
- преждевременная оптимизация — зло
- знать паттерны утечек







tg: @vzkhrv

Паттерны утечек на клиенте



Event Listeners

```
function handleClick () {
    const hugeObject = { ... }
    // ...
}

document.addEventListener('click', handleClick);
```



Event Listeners

- от событий нужно отписываться
- параметр `{once: true}`

```
function handleClick () {
    const hugeObject = { ... }
    // ...

    document.removeEventListener('click', handleClick);
}

document.addEventListener('click', handleClick, { once: true });
```



Detached DOM

```
let parent = document.getElementById("#parent");
let child = document.getElementById("#child");

parent.addEventListener("click", function(){
    child.remove();
});
```



Detached DOM

обнулять ссылку после использования

```
function removeChild() {
    if (child) {
        child.remove();
        child = null;
    }
}
parent.addEventListener("click", removeChild);
// ...
parent.removeEventListener("click", removeChild);
```



Detached window

```
let show = document.getElementById("#show");
let hide = document.getElementById("#hide");

let popup;

show.onclick = () => {
  popup = window.open('/some-url');
};

hide.onclick = () => {
  if (popup) { popup.close(); }
};
```



Detached window

```
let show = document.getElementById("#show");
let hide = document.getElementById("#hide");

let popup;

show.onclick = () => {
  popup = window.open('/some-url');
};

hide.onclick = () => {
  if (popup) { popup.close(); }
};
```



Detached window

```
let show = document.getElementById("#show");
let hide = document.getElementById("#hide");

let popup;

show.onclick = () ⇒ {
  popup = window.open('/some-url');
};

hide.onclick = () ⇒ {
  if (popup) { popup.close(); }
};
```



Detached window

```
let show = document.getElementById("#show");
let hide = document.getElementById("#hide");

let popup;

show.onclick = () => {
  popup = window.open('/some-url');
};

hide.onclick = () => {
  if (popup) { popup.close(); }
};
```



Detached window

```
let show = document.getElementById("#show");
let hide = document.getElementById("#hide");

let popup;

show.onclick = () ⇒ {
  popup = window.open('/some-url');
};

hide.onclick = () ⇒ {
  if (popup) { popup.close(); }
};
```



Detached window

использовать WeakRef для хранения ссылок на DOM-элементы

```
show.onclick = () ⇒ {
  popup = new WeakRef(window.open('/some-url'));
};

hide.onclick = () ⇒ {
  const win = popup.deref();
  if (win) { win.close(); }
};
```

